February 2021

CARRIE 2.0: INSIGHT EXTRACTION FROM UNSTRUCTURED DATA USING PROXIMITY CLUSTERING ENGINE

HP INC

Carrie 2.0: Insight Extraction from Unstructured Data Using Proximity Clustering Engine

Abstract

CaRRIE 2.0 extracts key insights/summaries from unstructured data such as papers, news document, notes, blog scrapped website, and RFP (Request for Proposal) data. The user gains faster understanding and insights to make informed decisions. The engine groups/ranks collections of documents based on keywords of interest presented within the documents. The user can then focus on a handful of documents aligned with their interests, reducing the time needed to read each document and create insights. He/she can also understand the larger data trends without needing to read the entire stack of documents, one by one. We accomplish those tasks by stacking and modifying natural language processing (NLP) algorithms, as well as creating a couple of new NLP algorithms. On the high-level, what we are doing is very similar to how a search engine algorithm works. Our solution is divided into three main parts: (i) Topic Modeling, (ii) Scoring and Ranking, and (iii) Insight extractions.

Keywords: Insight extractions, summarizations

Carrie 2.0

Introduction

Imagine the following scenario: Monday morning, you're assigned 10 lengthy proposal documents to read and derive insights from based on a list of provided questions. This task is due at weeks end. As an awesome employee, you read the 10 documents, identify and rank the documents which best answer each question, highlight the relevant information within the text, and deliver the answer to each question, just in time for Friday's deadline! The following Monday you're assigned the same task, but now with 123 lengthy proposal documents and the due date is still at weeks end. What are you going to do?! Can you (anyone?) complete the task on time and correctly answer/gather insights as well as rank those documents based on the questions?

Imagine on Tuesday, you not only deliver the list of answers, but provide even more insight, such as identifying the 5 most relevant proposal documents for certain questions. In addition, each relevant document is highlighted to identify statements aligned with each questions/keyword. Beyond the original assignment, you provide insight into the topics/requirements within the 123 proposals which weren't captured in the question set, hence the leadership team can plan and anticipate future trends.

We built an NLP insight engine to make this solution a reality. With this engine, we can:

1. Understand trends in large document collections, and therefore anticipate future trends
2. Rank/sort documents based on topics of interest
3. Gather insight from the most important documents
4. Derive insight beyond the original request
5. In the case of RFP documents:
    a. Identify RFPs closely aligned with defined feature set
    b. Prioritize resources towards deals with highest ranking, i.e. high probability of winning
    c. Enable consistent increased margins on those deals
    d. Inform product marketing on potential portfolio gaps

e. Feedback mechanism for marketing effort (do customers ask for the features we've invested in advertising?)

6. Process thousands of documents in less than an hour with high accurate result

This engine is divided into three main parts:

(i). Topic Modelling

(ii). Scoring and Ranking

(iii). Insight extraction

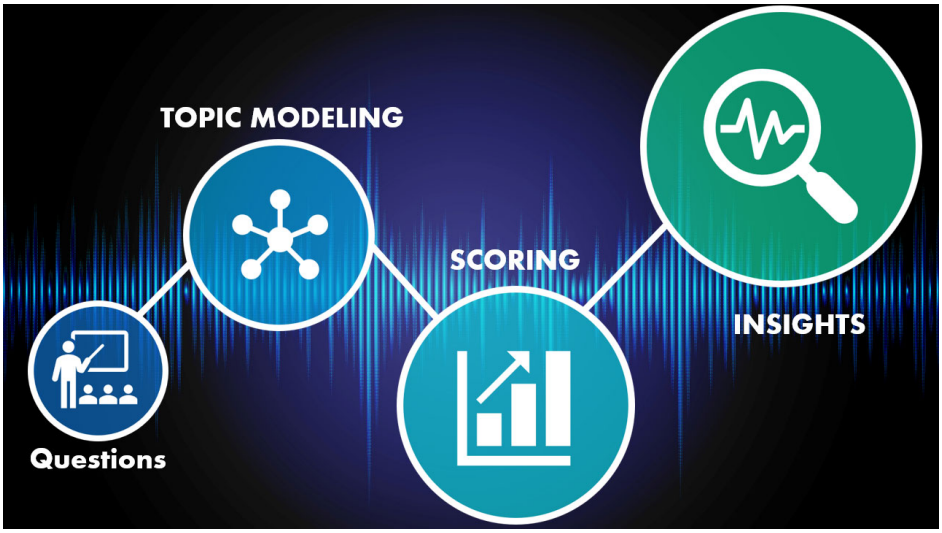

Figure 1: High level flow of the engine. Our solution begins with the questions, followed by topic modeling, scoring/ranking the documents, and finally deriving the insights from the top "n" documents.

Methodology and Results

We will discuss in detail each of these steps in the following section.

1. Topic Modeling

   Goal:

   a. Add weights to LDA model topic lexicons in order to highlight the features.
   b. Create a superset of lexicon from task corpus to include unique and rich features from corpus into topics

   Method:

   a. Data cleansing

      Real-world data is often incomplete, inconsistent, sometimes lacking certain behaviors/trends, and is likely to contain many errors. Hence the first step is to

prepare the data. Data processing involves converting raw, human readable data into a more relevant format, feature rich and effectively understood by algorithms or machines. Examples of data processing are shown below in Figure *2*. An example of data processing using the Gensim method is shown in Figure *3*.[1].

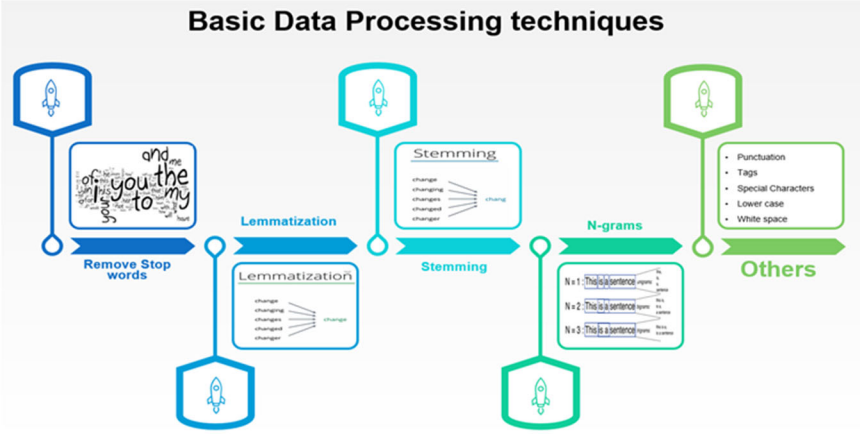

Figure 2: Techniques used to process the data.

```
from gensim.parsing.preprocessing import *
from nltk.stem.wordnet import WordNetLemmatizer

CUSTOM_FILTERS = [lambda x: x.lower(), strip_tags,strip_punctuation,strip_multiple_whitespaces,strip_numeric,
                  remove_stopwords,strip_short, stem_text]

s="hello how are you?  Hope you have a g00d DAY! I am add;ng random words ## but where running scoped amusement"
"amused amusing caring"
lemmatizer = WordNetLemmatizer()
print([lemmatizer.lemmatize(token) for token in preprocess_string(s, CUSTOM_FILTERS)])
```

```
['hello', 'hope', 'dai', 'add', 'random', 'word', 'run', 'scope', 'amus', 'amus', 'amus', 'care']
```

Figure 3: Data Processing example using Gensim

c. LDA topic modeling

Datasets vary widely and we cannot depend on labeled data or expected subjected matter expertise, therefore we used an unsupervised machine learning method, Topic Modeling. Our solution utilizes Latent Dirichlet Allocation (LDA) Topic Modeling, which is guided by two principles [2]: 1. Every document is mixture of topics 2. Every topic is mixture of words. The output of the LDA model provides insights of from the document corpus via topics and topic lexicons.

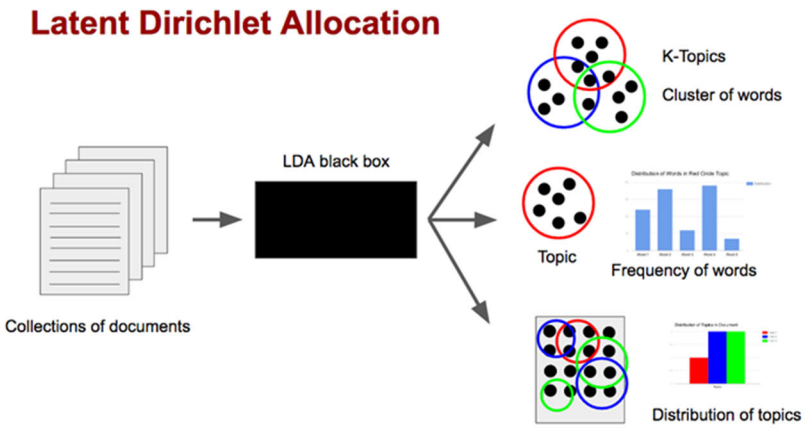

Figure 4: LDA Topic Modeling Method. Retrieved from: https://toolbox.kurio.co.id/topic-modeling-696d7ba2592f

d. Add weights to topic lexicons

LDA provides topic lexicons, which are ordered by relevancy to the topic, as shown in Figure *5*. This relevancy is then used to create weights for each word in the lexicon (Figure 6), as opposed to weighting each word equally in the next section (scoring).

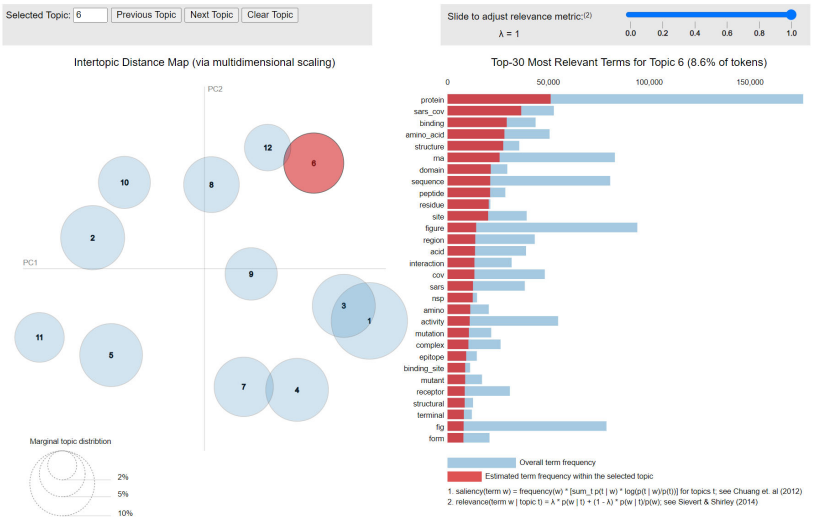

Figure 5: Visual representation of LDA modeling topics and ordered topic lexicon.

topic_id	topic_lexicon	lexicon_weight
topic_0	"cell protein fig virus figure expression viral membrane infected cell_line infection rna replication anti assay"	{'cell': 3, 'protein': 3, 'fig': 3, 'virus': 3, 'figure': 3, 'expression': 2, 'viral': 2, 'membrane': 2, 'infected': 2, 'cell_line': 2, 'infection': 1, 'rna': 1, 'replication': 1, 'anti': 1, 'assay': 1}
topic_1	"mers_cov sars_cov cov sars mers case patient infection human respiratory coronavirus transmission hcov camel calf"	{'mers_cov': 3, 'sars_cov': 3, 'cov': 3, 'sars': 3, 'mers': 3, 'case': 2, 'patient': 2, 'infection': 2, 'human': 2, 'respiratory': 2, 'coronavirus': 1, 'transmission': 1, 'hcov': 1, 'camel': 1, 'calf': 1}
topic_2	"health public_health disease infectious_disease public country risk data outbreak study research information surveillance health_care control"	{'health': 3, 'public_health': 3, 'disease': 3, 'infectious_disease': 3, 'public': 3, 'country': 2, 'risk': 2, 'data': 2, 'outbreak': 2, 'study': 2, 'research': 1, 'information': 1, 'surveillance': 1, 'health_care': 1, 'control': 1}
topic_3	"protein rna sequence amino_acid structure site domain binding residue virus figure region genome mutation interaction"	{'protein': 3, 'rna': 3, 'sequence': 3, 'amino_acid': 3, 'structure': 3, 'site': 2, 'domain': 2, 'binding': 2, 'residue': 2, 'virus': 2, 'figure': 1, 'region': 1, 'genome': 1, 'mutation': 1, 'interaction': 1}
topic_4	"virus sequence sample gene pcr strain specie genome analysis bat rna viral human primer study"	{'virus': 3, 'sequence': 3, 'sample': 3, 'gene': 3, 'pcr': 3, 'strain': 2, 'specie': 2, 'genome': 2, 'analysis': 2, 'bat': 2, 'rna': 1, 'viral': 1, 'human': 1, 'primer': 1, 'study': 1}
topic_5	"study infection influenza virus respiratory patient influenza_virus case child year age viral sample group clinical"	{'study': 3, 'infection': 3, 'influenza': 3, 'virus': 3, 'respiratory': 3, 'patient': 2, 'influenza_virus': 2, 'case': 2, 'child': 2, 'year': 2, 'age': 1, 'viral': 1, 'sample': 1, 'group': 1, 'clinical': 1}

Figure 6: LDA Lexicon with weights added for each word.

e. Task-Topic correlation algorithm

In the situation when we have Subject Matter Expert (SME), we can compare the lexicon we got from LDA to the list of lexicons coming from the total corpus. In that sense, we can enrich the total lexicons used for the next steps. In the other situation where we don't have the SME, we can combine the lexicon from the questions/task details and with the lexicons that are coming from the entire corpus. We will elaborate details about the second case.

In order to establish a better lexicon combination for a give task, we have introduced a new technique to enable complete perspective of the tasks and to not miss out any valuable information from the task themselves. As part of this technique, we used TFIDF along with a custom logic to extract prominent lexicons from the tasks and add them to the final list of lexicons [3]. The lexicons extracted from this technique will have higher weights added when compared with the lexicons extracted from the LDA modelling. This technique ensures the next steps in the algorithm will have a better performance.

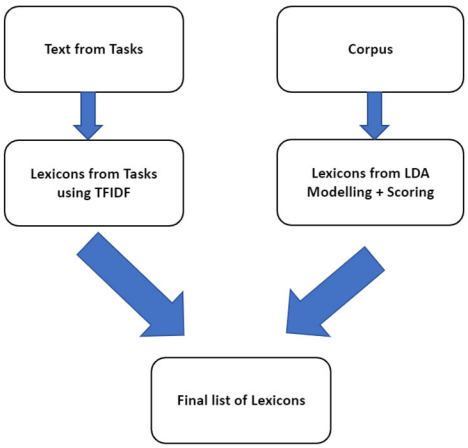

Figure 7: Flow to select the list of lexicon

2. Scoring and Ranking

Goal:

a. Score a set of documents with respect to weighted lexicons and have control over the score based on the lexicon length and weight.

Method:

Start with base algorithm of TFIDF (Term Frequency Inverse Document Frequency) to score the documents against a given set of lexicons. This works great but has some limitations, such as:

- Unbalanced result for fixed lexicon length
- Adjustment in score if lexicon has weights

An example of results from the base algorithm is shown in Figure 8a.

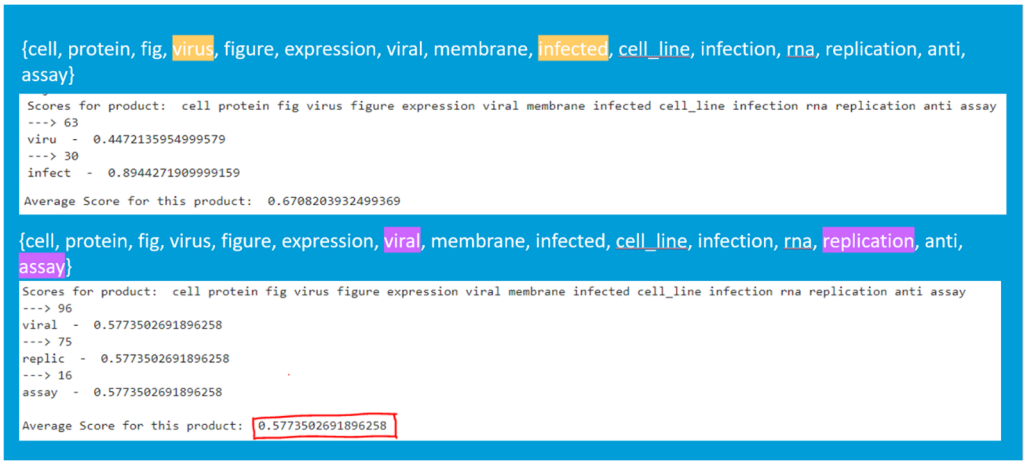

Figure 8a: Result from Adjusted Scoring method using weighted lexicon.

Our solution addresses these limitations by:

- *Normalizing the score* – Use lexicon size to normalize
- *Penalizing the score* – Calculate penalty factor by looking at weights

With this, the final adjusted scoring logic is:

Adjusted Score = (S/N) * (F/W)

where

F = Total Weight of words found in document

W = Total Lexicon weight

S = Total word score

N = Lexicon Size

An example of results from the Adjusted Score using a weighted lexicon is shown in Figure b.

Figure 8b: Result from Adjusted Scoring method using weighted lexicon.

Figure 8c shows the results from score and ranking algorithm. Each lexicon will have top n-documents that have the highest score.

	paper_id	task_id	score	word_score	topic_lexicon	lexicon_weight
0	555a36d1c3822baa8b6a2612b86ed5df5b1dbf7c	0	0.258199	{'virus': 0.2581988897471611, 'viral': 0.25819...	"age infection control viral clinical specific...	{'age': 1, 'infection': 3, 'control': 1, 'vira...
1	49b957bb681c7cae440ef4241a9fc99f37028d78	0	0.258199	{'virus': 0.2581988897471611, 'viral': 0.25819...	"age infection control viral clinical specific...	{'age': 1, 'infection': 3, 'control': 1, 'vira...
2	dcf0fae6dc17737b792734169c7d16468ab7a9dc	0	0.258199	{'virus': 0.2581988897471611, 'viral': 0.25819...	"age infection control viral clinical specific...	{'age': 1, 'infection': 3, 'control': 1, 'vira...
3	8c6ca63f974a55bcab6f773fc9cda85c938b93eb	0	0.258199	{'virus': 0.2581988897471611, 'viral': 0.25819...	"age infection control viral clinical specific...	{'age': 1, 'infection': 3, 'control': 1, 'vira...
4	8c43b5c8a56828ee5687a2b572dc05333c2815a0	0	0.258199	{'virus': 0.2581988897471611, 'viral': 0.25819...	"age infection control viral clinical specific...	{'age': 1, 'infection': 3, 'control': 1, 'vira...
0	85d9c134bccd818e9487193754d8c71a710161d6	1	0.230256	{'virus': 0.2672612419124244, 'treatment': 0.2...	"public infection data control patient health ...	{'public': 3, 'infection': 3, 'data': 2, 'cont...
1	5d12573335980 94fd630f16f989336a2c5d223a9	1	0.230256	{'virus': 0.2672612419124244, 'treatment': 0.2...	"public infection data control patient health ...	{'public': 3, 'infection': 3, 'data': 2, 'cont...
2	3c51c5a11684d15c9f3d39dc99b0fdf9f7a73847	1	0.230256	{'virus': 0.2672612419124244, 'treatment': 0.2...	"public infection data control patient health ...	{'public': 3, 'infection': 3, 'data': 2, 'cont...
3	ee0050c6fb81a4067d134010d0c80d21edb5df0b	1	0.230256	{'virus': 0.2672612419124244, 'treatment': 0.2...	"public infection data control patient health ...	{'public': 3, 'infection': 3, 'data': 2, 'cont...
4	d86146bbcb9ae18278150f4cb241ca1ce31fbf28	1	0.230256	{'virus': 0.2672612419124244, 'treatment': 0.2...	"public infection data control patient health ...	{'public': 3, 'infection': 3, 'data': 2, 'cont...
0	90d553a545579b2e27d0899c558e004ff33cd0f4	2	0.230256	{'virus': 0.2672612419124244, 'vaccine': 0.267...	"infection sample sequence human specific bind...	{'infection': 3, 'sample': 3, 'sequence': 3, '...
1	b7e7ff011d768680bee745105bd24389068a5a1f	2	0.230256	{'virus': 0.2672612419124244, 'vaccine': 0.267...	"infection sample sequence human specific bind...	{'infection': 3, 'sample': 3, 'sequence': 3, '...

Figure 8c: Result table for each questions/task. Each questions/task will have top 5 documents that will have the highest score.

3. Insight Engine

Goal:

a. Extract the insights and summaries from top *n* ranked documents per topic.

Method:

The flow of the insight engine is shown in Figure .

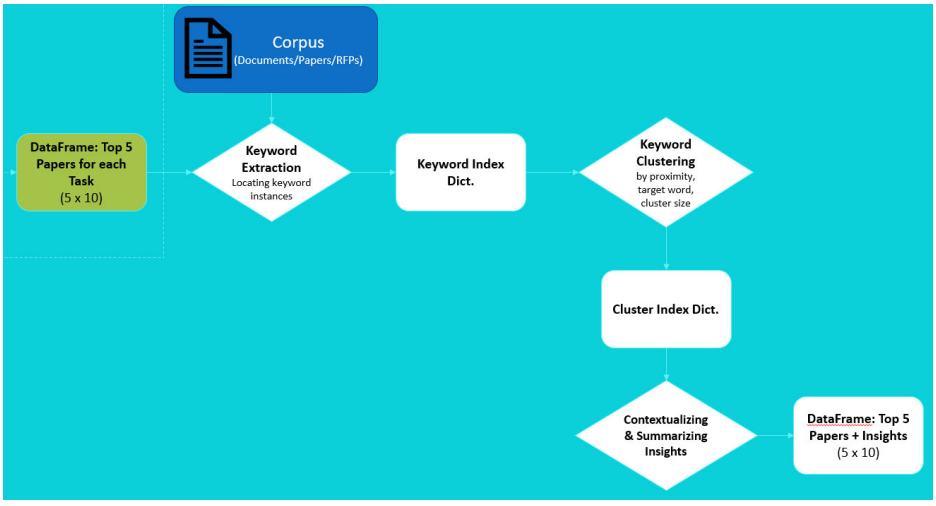

Figure 9: Insight engine flow diagram

We started by getting the top *n*-documents, as shown in the figure above. Next, we proceeded through the following steps:

1. Cleanse the text data, such as removing non-text related characters.
2. Tokenize and steam each word in the text.
3. Create indexing (position of each word in the text) and locate indexes for each keyword of interest:
4. Combine indexes into one sorted list and cluster keywords within a moving window of size "m", which scans the sorted list of keyword instances:
5. To validate each moving window, multiple criteria must be met. Here we define the criteria parameters:
    a) The number of unique keywords required within each moving window. This threshold can range from 1 to *m* and will be used for all moving windows. For example, if the minimum threshold is 3 and the moving window in question is:

        ['security', 'security', 'network', 'network']

        There are 2 unique keywords ('security' and 'network'). This moving window would be ignored. However, for a moving window of:

        ['security', 'network', 'monitor', 'security']

        There are 3 unique keywords ('security', 'network', 'monitor') and this moving window meets the first criterion.
    b) Target keyword(s): At least one target keyword must be present within the window to determine the relevance to the question. For example, if we want to generate insights around target words ['security', 'detection', 'vulnerability'], the moving window must include *at least one* of these target words. If not, the moving window is ignored.
    c) The maximum proximity of keywords within the moving window. This is measured by the standard deviation of index values within a window, for which the proximity parameter σ_{max} is defined to limit the range of a cluster. For example, the moving window indexes:

        [2, 100, 350, 390],

        have a proximity (standard deviation) of 163.8, indicating the keywords are relatively spread out. In contrast, the moving window:

        [300, 302, 304, 315],

has a proximity of 5.81, indicating the keywords are very near. Setting this standard deviation threshold too low will result in very few valid clusters, while too high may result in far too many overlapping clusters to be insightful. Only moving windows with proximity values < σ_{max} are kept.

6. Return the minimum and maximum index values of the validated clusters to retrieve these sections of the text.

The cluster indexes are used to generate two views of the document:

1. Contextualized View in which the original paper is shown with insight clusters highlighted
2. Summarization View which extracts key sentences from the document using a cosine similarity algorithm

An example of the Context View output of the insight engine results can be seen in Figure 7.

Figure 7: Context View, which highlights text within original document. Words from topic lexicon are outlined in red to enhance presentation (*note: outlines are not part of the actual output*).

Given the Topic Lexicon, the insight engine uses a cosine similarity algorithm to summarize the important sections of the document that are relevant to the lexicon. We can further refine the results into list of highlighted summaries.

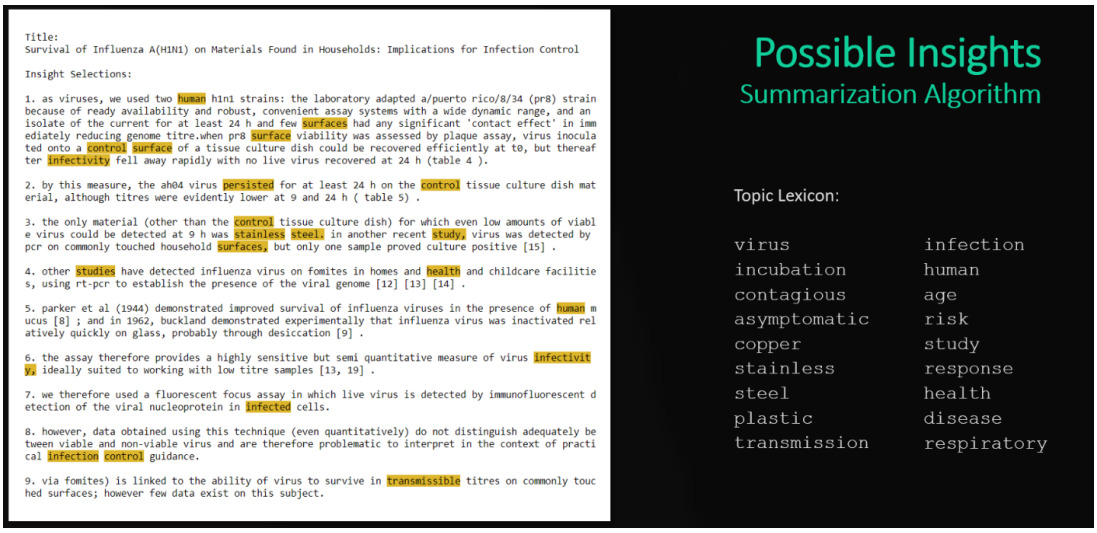

Figure 8: Summarization View with highlighted topic lexicon words.

The final output of the Insight Engine stores these highlighted results in a tabular format (DataFrame) for each of the top *n* documents per task.

paper_id	task_id	score	word_score	topic_lexicon	lexicon_weight	full_text_insights	summary_insights
555a36d1c3822baa8b6a2612b86ed5df5b1dbf7c	0	0.258199	{'virus': 0.2581988897471611, 'viral': 0.25819...	"response information control infection human ...	{'response': 1, 'information': 1, 'control': 1...	Title:\nEnhanced inflammation in New Zealand w...	Title:\nEnhanced inflammation in New Zealand w...
49b957bb681c7cae440ef4241a9fc99f37028d78	0	0.258199	{'virus': 0.2581988897471611, 'viral': 0.25819...	"response information control infection human ...	{'response': 1, 'information': 1, 'control': 1...	Title:\nProphylactic and Therapeutic Efficacy ...	Title:\nProphylactic and Therapeutic Efficacy ...
dcf0fae6dc17737b792734169c7d16468ab7a9dc	0	0.258199	{'virus': 0.2581988897471611, 'viral': 0.25819...	"response information control infection human ...	{'response': 1, 'information': 1, 'control': 1...	<Title Not Available> \n\n INTRODUCTION (URD) ...	<Title Not Available>\n\nSummarized Selections...
8c6ca63f974a55bcab6f773fc9cda85c938b93eb	0	0.258199	{'virus': 0.2581988897471611, 'viral': 0.25819...	"response information control infection human ...	{'response': 1, 'information': 1, 'control': 1...	Title:\nSickle-cell disease in febrile childre...	Title:\nSickle-cell disease in febrile childre...
8c43b5c8a56828ee5687a2b572dc05333c2815a0	0	0.258199	{'virus': 0.2581988897471611, 'viral': 0.25819...	"response information control infection human ...	{'response': 1, 'information': 1, 'control': 1...	<Title Not Available> \n\n Introduction The 20...	<Title Not Available>\n\nSummarized Selections...

Figure 9: Insights DataFrame with Contextualized Full Text and Summary views.

Summary

We have built an NLP algorithm to find the closest answers within a document or a stack of document given a list of questions (keywords). Think about when you read a chapter in a book and then by the end of the chapter you will get a list of questions about the chapter you just read. Using this algorithm, one can find the closest answers to those questions. Note that this algorithm has limitations, for example if the questions contains some deduction then the algorithm cannot get the right answer. This algorithm is a completion as well as an enhancement from the previous algorithm we built called Carrie 1.0 [4].

References

[1] Radim Řehůřek and Petr Sojka (2010). Software framework for topic modelling with large corpora. Proc. LREC Workshop on New Challenges for NLP Frameworks.

[2] Blei, David M.; Ng, Andrew Y.; Jordan, Michael I (January 2003). Lafferty, John (ed.). "Latent Dirichlet Allocation". *Journal of Machine Learning Research*. 3 (4–5): *pp.* 993–1022. doi:10.1162/jmlr.2003.3.4-5.993. Archived from the original on 2012-05-01. Retrieved 2006-12-19.

[3] Rajaraman, A.; Ullman, J.D. (2011). "Data Mining" (PDF). *Mining of Massive Datasets*. pp. 1–17. doi:10.1017/CBO9781139058452.002. ISBN 978-1-139-05845-2

[4] Wiranata, Anton; Bansal, Jai. "Critical Request for Proposals (RFPs) Requirement Insight Engines (CaRRIE). Tdcommons.org, volume 2688, 2019-11-14.

Disclosed by Katy Ferguson, Manjunath Bhuyar, Sheela Choudhari,

Vijay Reddy, Jeremiah Medina and Anton Wiranata, HP Inc.